日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年 ５月１８日

CERTIFIED COPY OF PRIORITY DOCUMENT

出願番号
Application Number: 特願２０００－１５２６６５

出願人
Applicant(s): 株式会社日立製作所

App. No. 09/855,681
(703) 684-1120
MATTINGLY, STANGER & MALUR, P.C.
1800 DIAGONAL ROAD
SUITE 370
ALEXANDRIA, VIRGINIA 22314
Dkt. No. ASA-1000

BEST AVAILABLE COPY

２００１年 ４月２０日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造

【書類名】　特許願
【整理番号】　K00000391
【提出日】　平成12年 5月18日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 11/28
【請求項の数】　7
【発明者】
【住所又は居所】　神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内
【氏名】　鶴ケ崎　俊博
【特許出願人】
【識別番号】　000005108
【氏名又は名称】　株式会社日立製作所
【代理人】
【識別番号】　100075096
【弁理士】
【氏名又は名称】　作田　康夫
【手数料の表示】
【予納台帳番号】　013088
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1
【物件名】　図面　1
【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　マルチプロセスの表示方法及び装置並びに表示プログラムを格納した記録媒体

【特許請求の範囲】

【請求項１】

プログラムを実行するマルチプロセッサの実行状態を表示するための表示方法であって、前記プログラムを構成するプロセスを所定の図形で表示し、前記プロセス間の関係を示すよう前記図形間に所定の表示を行い、前記プロセスの動作状態に連動して前記図形を特徴表示することを特徴とするマルチプロセスの表示方法。

【請求項２】

請求項１記載のマルチプロセスの表示方法であって、前記プロセス間の関係は、プロセス間の親子関係又は兄弟関係であることを特徴とするマルチプロセスの表示方法。

【請求項３】

請求項１記載のマルチプロセスの表示方法であって、前記プロセスの動作状態は、プロセスの生成・開始・再開・停止・終了の少なくとも１つであることを特徴とするマルチプロセスの表示方法。

【請求項４】

請求項１記載のマルチプロセスの表示方法であって、各プロセスの詳細情報を前記図形と同一画面に表示することを特徴とするマルチプロセスの表示方法。

【請求項５】

請求項４記載のマルチプロセスの表示方法であって、前記図形の特徴表示と前記各プロセスの詳細情報とは、連動して強調表示されることを特徴とするマルチプロセスの表示方法。

【請求項６】

プログラムを実行するマルチプロセッサの実行状態を表示するための表示装置であって、前記プログラムを構成するプロセスを所定の図形で表示し、前記プロセス間の関係を示すよう前記図形間に所定の表示を行う第一の表示手段と、前記

プロセスの動作状態に連動して前記図形を特徴表示する第二の表示手段とを備えたことを特徴とするマルチプロセスの表示装置。

【請求項７】

プログラムを構成するプロセスを所定の図形で表示し、前記プロセス間の関係を示すよう前記図形間に所定の表示を行い、前記プロセスの動作状態に連動して前記図形を特徴表示することによりプログラムを実行するマルチプロセッサの実行状態を表示することを特徴とするマルチプロセスの表示プログラムを格納した記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、マルチプロセスの表示方法及び装置並びに表示プログラムを格納した記録媒体に係り、特にプログラムを構成するプロセスの動作状態の変化を、プログラムの動作と連動しながら視覚的な観点でデバッグするためのマルチプロセスプログラムのデバッグ方法及び装置に関するものである。

【０００２】

【従来の技術】

従来からデバッグを支援するために様々な方法が考えられている。例えば、特開平５－２３３３２３号においては、並列プログラムのデバッグを対話的に効率良く行うためのデバッグ方法が開示されている。また、特開平８－１８５３４０号においては、並列プログラムを可視化するためのツールである並列プログラム可視化方法が開示されている。しかし、これらの従来技術では、並列プログラムを構成するプロセスの親子、及び兄弟関係を視覚的に特徴表示する機能は有していない。また、プロセスの生成、開始、再開、停止、及び終了というプロセスの動作状態を特徴表示する機能は有していない。尚、特開平１１－１６１５１４号においては、複数のプロセスで構成させるプログラムのデバッグ方法を示しているが、プログラムの動作と連動しながら、プログラムを構成するプロセスのデバッグを行える機能は有していない。

【０００３】

【発明が解決しようとする課題】

上記従来技術では、並列プログラムを構成するプロセスの動作状態の変化に応じたデバッグ方法が配慮されておらず、例えば、並列プログラムを構成するプロセスが増大した場合、プロセスの生成、開始、再開、停止、及び終了のようなプロセスの動作に着目してプログラムのデバッグを行うことが困難という問題があった。また、並列プログラムの動作と連動したデバッグ方法が配慮されておらず、例えば、動作プロセスと、終了プロセスとの親子、及び兄弟関係も重要なデバッグ作業を行うための情報（以下、デバッグ情報と呼ぶ）となるが、そのデバッグ情報に着目したデバッグが困難という問題があった。

【０００４】

本発明の目的は、マルチプロセスプログラムのデバッグを効率的に行うことを可能とするマルチプロセスの表示方法及び装置並びに表示プログラムを格納した記録媒体を提供することにある。

【０００５】

【課題を解決するための手段】

上記目的は、マルチプロセスプログラムを構成する複数のプロセスの動作状態に関連付けたデバッグ情報を、プログラムの動作と連動しながら特徴表示する点、また動作プロセスと終了プロセスとの関係を視覚的に区別表示する点に着目することにより達成される。このような点に着目することにより、複数のプロセスで構成されるマルチプロセスプログラムのデバッグを効果的に可能とするデバッグ支援方法及び装置を提供できる。

【０００６】

より具体的には、プログラム（マルチプロセスプログラム、並列プログラムを含む）を実行するマルチプロセッサの実行状態を表示するための表示方法であって、前記プログラムを構成する複数のプロセスをそれぞれ所定の図形（例えば、矩形）で表示（以下、プロセスボックスとも呼ぶ）し、前記プロセス間の関係を示すよう前記図形間に所定の表示（例えば、線による連結表示等の視覚的に表現する表示）を行い、前記プロセスの動作状態に連動して前記図形を特徴表示（例

えば、プロセスボックスの枠線を、動作プロセスは実線、終了プロセスは破線）することで、プロセスの動作状態を視覚的に表現したもの（以下、プロセス関連グラフとも呼ぶ）により達成する。ここで、プロセス間の関係とは、プロセスの親子又は兄弟関係であっても良い。尚、プロセスの動作状態とは、例えばプロセスの生成・開始・再開・停止・終了等を示す。また、プロセス関連グラフは、プログラムの動作に連動しながら、プロセスボックスの特徴表示状態を変化させるものであっても良い。さらに、プロセスの詳細な動作状態（プログラム名称やソースコード位置の情報）を簡略化して知ることが出来るようにするために、プロセス関連グラフと、プロセスの詳細な動作情報（以下、プロセス詳細情報と呼ぶ）を同一画面に表示し、プロセス関連グラフとプロセス詳細情報は、どちらか片方の情報から、あるプロセスの強調表示操作を行った場合でも、プロセス関連グラフと、プロセス詳細情報の両情報のプロセス情報を連動して強調表示するようにしたものであっても良い。

【０００７】

尚、上記目的を達成するためには、上述した機能を実現するプログラム若しくはプログラムを格納した記録媒体であっても良い。

【０００８】

【発明の実施の形態】

以下、本発明の一実施形態を図面に基づいて詳細に説明する。

【０００９】

図１は、マルチプロセスプログラムのデバッグ方法の構成図である。デバッグ作業を行うプログラム（以下、デバッグプログラムと呼ぶ）１００、デバッガシステム部１１０、及び表示装置１５０で構成される。デバッグプログラム１００は、プログラムを構成する複数のプロセス群（以下、プロセスをデバッグプロセスと呼ぶ）１０１で構成される。デバッガシステム部１１０は、デバッグプログラム１００の実行を制御、及び実行を監視するデバッガエンジン部１２０と、デバッグプログラム１００の動作状態を、表示装置１５０に表示するデバッガ情報表示部１４０で構成させる。デバッガエンジン部１２０は、デバッグプログラム１００を、デバッグプロセス１０１単位に実行を制御、及び実行を監視するデバ

ッグ実行処理部１２１と、デバッガエンジン部１２０より、デバッガ情報表示部１４０へ、デバッグプロセス１０１の動作状態を伝達するために使用するプロセス動作情報１３１を、プロセス動作情報ファイル１３０に出力するプロセス動作情報出力部１２２で構成される。デバッグプロセス１０１の動作状態に変化が生じた場合、デバッグ実行処理部１２１が検知し、プロセス動作情報出力部１２２に制御を移す。プロセス動作情報出力部１２２は、プロセス動作情報ファイル１３０に、プロセス動作情報１３１を出力する。プロセス動作情報１３１の詳細は図２で説明する。プロセス動作情報監視部１４１は、デバッグプロセス１０１の動作に連動しながら、表示装置１５０の表示状態を変化させるために、プロセス動作情報ファイル１３０へのプロセス動作情報１３１出力の有無を調べる。プロセス動作情報１３１が出力されている場合、プロセス動作情報監視部１４１は、プロセス動作情報取得部１４２へ制御を移し、プロセス動作情報取得部１４２では、プロセス動作情報ファイル１３０から、プロセス動作情報１３１を入力し、プロセス情報テーブル組立部１４３へ制御を移す。プロセス情報テーブル組立部１４３は、入力したプロセス動作情報１３１から、デバッグプロセス１０１を管理するプロセス情報テーブルを作成及び更新し、プロセス情報表示部１４４に制御を移す。プロセス情報テーブルの詳細は図３で説明する。プロセス情報表示部１４４は、プロセス関連グラフ表示部１４５と、プロセス詳細情報表示部１４６で構成され、プロセス情報テーブル組立部１４３で作成、及び更新されたプロセス情報テーブルを参照し、プロセス関連グラフ１５１とプロセス詳細情報１５２を、表示装置１５０に表示する。

【００１０】

図２は、プロセス動作情報１３１のデータ構成図である。プロセス動作情報１３１は、プロセス状態情報２０１と、プロセス詳細情報２０２で構成される。プロセス状態情報２０１は、デバッグプロセス１０１の動作状態情報を管理し、プロセス詳細情報２０２は、デバッガエンジン部１２０が、デバッグプロセス１０１を、内部的に管理するために付加する番号（以下、ノード番号と呼ぶ）や、プロセス番号、プログラム名称、及び停止位置のソースファイル情報など、プロセス状態情報２０１に応じた情報を管理する。プロセス動作情報２１０は、デバッ

グプロセス１０１が新たに生成された場合のプロセス動作情報１３１の出力例である。ノード番号（０）２１２、プロセス番号（１０）２１３のデバッグプロセスから、ノード番号（１）２１４、プロセス番号（２０）２１５のデバッグプロセスが生成されたことを意味する。プロセス動作情報２２０は、ファイルの動作が開始された場合のプロセス動作情報１３１の出力例である。ノード番号（１）２２２、プロセス番号（２０）２２３のデバッグプロセスで、プログラム名称（Ｐｒｏｇ２＿０）２２４の実行が開始されたことを意味する。プロセス動作情報２３０は、ブレークポイントなどの要因により、動作を停止した場合のプロセス動作情報１３１の出力例である。ノード番号（２）２３２、プロセス番号（２１）２３３のデバッグプロセスは、ソースファイル（ｆｕｎｃｔｉｏｎ．ｃ）２３４の行番号（１４６行目）２３５で停止したことを意味する。プロセス動作情報２４０は、デバッグプロセスの動作を再開した場合のプロセス動作情報１３１の出力例であり、ノード番号（０）２４２、プロセス番号（１０）２４３のデバッグプロセスは、動作を再開したことを意味する。プロセス動作情報２５０は、デバッグプロセスの動作が終了した場合のプロセス動作情報１３１の出力例であり、ノード番号（３）２５２、プロセス番号（３０）２５３のデバッグプロセスは、動作が終了したことを意味する。

【００１１】

図３は、図１に示すプロセス情報テーブル組立部１４３により作成、及び更新され、プロセス関連グラフ表示部１４５、及びプロセス詳細情報表示部１４６で参照するプロセス情報テーブル３００の詳細である。デバッグプロセス情報３１０、プロセスボックス座標位置情報３４０、子供デバッグプロセス情報３２０、兄弟デバッグプロセス情報３３０で構成される。デバッグプロセス情報３１０は、ノード番号情報３１１、プロセス番号情報３１２、プログラム名称情報３１３、及び動作状態情報３１４を管理する。プロセスボックス座標位置情報３４０は、プロセス情報テーブル３００で管理しているデバッグプロセスを、プロセスボックスとしてプロセス関連グラフに、表示した場合の座標情報として、プロセスボックスのｘ軸情報３４１、ｙ軸情報３４２、幅情報３４３、及び高さ情報３４４を管理する。子供デバッグプロセス情報３２０は、プロセス情報テーブル３０

０で管理しているデバッグプロセスより生成されたデバッグプロセス（以下、子プロセスと呼ぶ）のプロセス情報テーブルポインタ３２１を管理し、兄弟デバッグプロセス情報３３０は、プロセス情報テーブル３００で管理しているデバッグプロセスと、兄弟の関係にあるデバッグプロセス（以下、兄弟プロセスと呼ぶ）のプロセス情報テーブルポインタ３３１を管理する。

【００１２】

図４は、プロセス動作情報ファイル１３０からプロセス動作情報１３１を入力し、表示装置１５０に表示するデバッガ情報表示部１４０の処理手順をフローチャートにしたものである。まず、プロセス動作情報監視部１４１は、プロセス動作情報ファイル１３０に、プロセス動作情報１３１が出力されているかを調べる（ステップ４１１）。プロセス動作情報１３１が出力されている場合、プロセス動作情報取得部１４２に制御を移す。プロセス動作情報取得部１４２は、プロセス動作情報ファイル１３０から、プロセス動作情報１３１を入力し（ステップ４２１）、プロセス情報テーブル組立部１４３へ制御を移す。プロセス情報テーブル組立部１４３では、プロセス動作情報１３１に管理されているプロセス状態情報より、デバッグプロセスの動作状態を調べる（ステップ４３１）。プロセス状態情報が、デバッグプロセス生成時は、プロセス情報テーブル作成処理（ステップ４３２）、デバッグプロセスの動作開始、再開、停止、及び終了時は、プロセス情報テーブル更新処理（ステップ４３３）を行い、プロセス情報表示部１４４に制御を移す。プロセス情報テーブル作成処理４３２の詳細な実施例は図５に、プロセス情報テーブル更新処理４３３の詳細な実施例は図６に記載する。プロセス情報表示部１４４は、プロセス情報テーブルをキーにして、プロセス関連グラフ表示部１４５で、プロセス関連グラフ表示用のデータ作成処理４４２を、プロセス詳細情報表示部１４６で、プロセス詳細情報表示用のデータ作成処理４４３をデバッグプロセスの数分繰り返し処理し（ステップ４４１～４４４）、表示装置１５０に制御を移す。プロセス関連グラフ表示データ作成処理４４２、及びプロセス詳細情報表示データ作成処理４４３の詳細な実施例は図７に記載する。

【００１３】

図５は、プロセス情報テーブル作成処理４３２の詳細な実施例である。図２で

説明したプロセス動作情報２１０を入力例として説明する。まず、新しいデバッグプロセス１０１を生成したデバッグプロセス（入力例の場合、ノード番号０～２１２、プロセス番号１０～２１３のデバッグプロセス。以下、親プロセスと呼ぶ）を管理しているプロセス情報テーブルの検索５０１を行う。検索方法は、親プロセスのノード番号（入力例の場合、ノード番号０～２１２）と、プロセス情報テーブルに管理しているノード番号情報との一致判定５０２で行う。親プロセスを管理しているプロセス情報テーブルを検索できたら、新しく生成されたデバッグプロセス（入力例の場合、ノード番号１～２１４、プロセス番号２０～２１４のデバッグプロセス。以下、生成プロセスと呼ぶ）のプロセス情報テーブルを作成５０３し、親プロセスのプロセス情報テーブルで管理している子供デバッグプロセス情報に登録する。同時に、兄弟デバッグプロセス情報も登録し、プロセス情報表示部～１４４に制御を移す。

【００１４】

図６は、プロセス情報テーブル更新処理４３３の詳細な実施例である。図２のブレークポイントなどの要因により、動作を停止した場合のプロセス動作情報２３０を入力例として説明する。まず、動作の停止したデバッグプロセス（入力例の場合、ノード番号２～２３２、プロセス番号２１～２３３のデバッグプロセス）を管理しているプロセス情報テーブルの検索６０１を行う。検索方法は、デバッグプロセスのノード番号（入力例の場合、ノード番号２～２３２）と、プロセス情報テーブルに管理しているノード番号情報との一致判定５０２で行う。デバッグプロセスを管理しているプロセス情報テーブルを検索出来たら、デバッグプロセスの動作状態を、プロセス動作情報２３０のプロセス状態情報２３１で判定し（ステップ６０３）、入力例のプロセス動作情報２３０のプロセス状態情報２３１は動作停止を表しているため、デバッグプロセスを管理しているプロセス情報テーブルの動作状態情報に、動作停止情報を設定し（ステップ６０６）、プロセス情報表示部１４４に制御を移す。同様に、プロセス動作情報２２０を入力例とした場合、動作状態情報に動作開始情報を設定（ステップ６０４）、プロセス動作情報２４０を入力例とした場合、動作状態情報に動作再開情報を設定（ステップ６０５）、プロセス動作情報２５０を入力例とした場合、動作状態情報に動

作終了情報を設定し（ステップ６０７）、プロセス情報表示部１４４に制御を移す。

【００１５】

図７は、プロセス情報表示部１４４の詳細な実施例である。まず、プロセス情報テーブル組立部１４３で作成、及び更新された、プロセス情報テーブルを入力し（ステップ７０１）、全てのデバッグプロセスのデバッグ情報表示処理が完了したかを判断する（ステップ７０２）。全てのデバッグプロセスのデバッグ情報表示処理が完了した場合、表示装置１５０へ制御を移す。プロセス関連グラフ表示部１４５では、デバッグプロセスをプロセスボックスとして表示するために、プロセスボックスの表示位置情報（ｘ軸、ｙ軸、幅、及び高さ）の計算を行い（ステップ７０３）、プロセス情報テーブルのプロセスボックス座標位置情報に、それぞれの計算値を設定する（ステップ７０４）。その後、デバッグプロセスの動作状態（生成、開始、再開、停止、及び終了）と連動してプロセスボックスを特徴表示するため、プロセス情報テーブルの動作状態情報を参照し（ステップ７０５）、動作状態に応じたプロセスボックス表示用画面データの作成（ステップ７１０、ステップ７１１、ステップ７１２、ステップ７１３、又はステップ７１４）を行い、プロセス詳細情報表示部１４６へ制御を移す。プロセス詳細情報表示部１４６は、プロセス詳細情報表示用の画面データの作成を行う（ステップ４４３）。

【００１６】

図８は、表示装置１５０の同一画面に表示されたプロセス関連グラフ１５１と、プロセス詳細情報１５２の表示例である。まず、プロセス関連グラフ１５１に表示されるプロセスボックス（８０１、８０２、８０３、及び８０４）には、デバッグプロセスを示す情報として、プロセス番号（ノード番号）のデバッグプロセス情報を表示する。例として、プロセスボックス８０１のデバッグプロセス情報８１０は、プロセス番号１０（ノード番号０）のデバッグプロセスである事を示す。更に、プロセスボックス８０１は、プロセスボックス８０２と線により連結表示８１１していることで、プロセスボックス８０２で示されるデバッグプロセスを生成した親プロセスとなる。また、プロセスボックス８０４とも、線によ

り連結表示８１２していることで、プロセスボックス８０４で示されるデバッグプロセスを生成した親プロセスとなる。これで、プロセスボックス８０２と、プロセスボックス８０４間は、兄弟プロセスであることが分かる。プロセスボックス８０２は、プロセスボックス８０３と線により連結表示８１３されていることで、プロセスボックス８０３で示されるデバッグプロセスを生成した親プロセスである。プロセスボックス８０３、及びプロセスボックス８０４は、線により連結表示されているプロセスボックスが無いため、子プロセスの生成は行っていない。更に、プロセスボックス８０１のデバッグプロセスは、プロセスボックスの枠線を太線により特徴表示８４１していることで、動作再開状態のデバッグプロセスであることを示し、プロセスボックス８０２と、プロセスボックス８０４のプロセスは、プロセスボックスの枠線を細線により特徴表示（８４２、８４３）していることで動作停止状態、プロセスボックス８０３のプロセスは、プロセスボックスの枠線を点線により特徴表示８４３していることで実行終了状態と、デバッグプロセスの動作状態に連動した表示を行う。プロセス詳細情報１５２は、行の情報として、デバッグプロセスの詳細な動作情報（８２１、８２２、８２３、及び８２４）を表示し、列の情報として、ノード番号８３１、プログラム名称８３２、プロセス番号８３３、動作状態８３４、及び停止位置８３５のデバッグ情報を表示する。

【００１７】

図９は、プロセス関連グラフ９１１からの、特定デバッグプロセスのデバッグ情報選択状態（以下、強調表示と呼ぶ）操作９１０と、プロセス詳細情報９２１からの、特定デバッグプロセスのデバッグ情報強調表示操作９２０の手順である。プロセス関連グラフ９１１からの強調表示操作９１０では、マウスなどにより、プロセス関連グラフ９１１から、プロセス番号２０（ノード番号１）のデバッグプロセスのデバッグ情報強調表示操作９１２が行われた場合、マウスのポイント位置と、プロセス情報テーブルのプロセスボックス座標位置情報で管理している座標情報から、マウスのポイント位置に、プロセスボックスとして表示されているデバッグプロセスを管理しているプロセス情報テーブルを検索する（ステップ９１３）。検索したプロセス情報テーブルより、強調表示操作の行われたデバ

ッグプロセスのノード番号は１であることを取得でき（ステップ９１４）、取得したノード番号をキーにして、プロセス詳細情報９１６に表示しているプロセス詳細情報から、ノード番号の一致するプロセス詳細情報の行の情報を検索する（ステップ９１５）。プロセス情報テーブル検索処理（ステップ９１３）で検索したプロセス情報テーブルのプロセスボックス座標位置情報から、プロセスボックスの強調表示処理を行い（ステップ９３０）、プロセス詳細情報検索（ステップ９１５）で検索した行の情報を基に、プロセス詳細情報の強調表示処理を行う（ステップ９３１）。これで、プロセス関連グラフ９４１のノード番号１のデバッグプロセスを示すプロセスボックス９４３、及びプロセス詳細情報９４２のノード番号１のデバッグプロセスを示すプロセス詳細情報９４４が連動して強調表示され、表示装置９４０に表示する。次に、プロセス詳細情報９２１からの強調表示操作９２０では、マウスなどにより、プロセス詳細情報９２１から、プロセス番号２０、ノード番号１のデバッグプロセスのデバッグ情報強調表示操作９２２が行われた場合、マウスのポイント位置から、行の情報としてデバッグプロセスの詳細情報を表示している文字列の取得を行い（ステップ９２３）、取得した文字列からノード番号を取得する（ステップ９２４）。取得したノード番号をキーにして、強調表示操作の行われたデバッグプロセスを管理している、プロセス情報テーブルを検索する（ステップ９２５）。プロセス関連グラフからの強調表示操作９１０と同様、プロセス関連グラフ９４１のノード番号１のデバッグプロセスを示すプロセスボックス９４３、及びプロセス詳細情報９４２のノード番号１のデバッグプロセスを示すプロセス詳細情報９４４が連動して強調表示され、表示装置９４０に表示される。

【００１８】

このように本実施例によれば、マルチプロセスプログラムを構成する複数のプロセスの動作状態に関連付けたデバッグ情報を、プログラムの動作と連動しながら特徴表示することにより、プロセスの生成、開始、再開、停止、及び終了のような、プロセスの動作に着目したプログラムのデバッグを行うことができる。また、プロセス関連グラフと、プロセス詳細情報を同一画面に表示し、プロセス関連グラフとプロセス詳細情報は、連動して強調表示することにより、プロセスの

詳細な動作状態（プログラム名称やソースコード位置の情報）を簡略化して知ることが出来る。

【００１９】

【発明の効果】

本発明によれば、各プロセッサにおけるマルチプロセスプログラムの実行状態の詳細情報を可視化することができ、デバッグの効率化を支援することができる。

【図面の簡単な説明】

【図１】

マルチプロセスプログラムのデバッグ方法の構成図である。

【図２】

プロセス動作情報のデータ構成図である。

【図３】

プロセス情報テーブル詳細である。

【図４】

デバッガ情報表示部の処理手順をフローチャートにしたものである。

【図５】

プロセス情報テーブル作成処理の詳細な実施例である。

【図６】

プロセス情報テーブル更新処理の詳細な実施例である。

【図７】

プロセス情報表示部の詳細な実施例である。

【図８】

表示装置に表示されたプロセス関連グラフと、プロセス詳細情報の表示例である。

【図９】

プロセス関連グラフとプロセス詳細情報が連動して強調表示される手順である。

【符号の説明】

１００　デバッグプログラム

１１０　デバッガシステム部

１２０　デバッガエンジン部

１３０　プロセス動作情報ファイル

１４０　デバッガ情報表示部

１５０　表示装置

２０１～２５３　プロセス動作情報出力例

３００　プロセス情報テーブル

３１０　デバッグプロセス情報

３２０　プロセスボックス座標情報

３３０　子供デバッグプロセス情報

３４０　兄弟デバッグプロセス情報

４１１～４４４　デバッガ情報表示部の処理手順

５０１～５０４　プロセス情報テーブル作成処理

６０１～６０７　プロセス情報テーブル更新処理

７０１～７１４　プロセス情報表示部

８０１～８０４　プロセスボックス

８１０　デバッグプロセス情報

８１１～８１３　プロセス親子及び兄弟関係の特徴表示

８２１～８２４　デバッグプロセスの詳細な動作情報（行情報）

８３１～８３５　デバッグプロセスの詳細な動作情報（列情報）

８４１～８４４　プロセスボックスの特徴表示

９１０　プロセス関連グラフからの強調表示操作

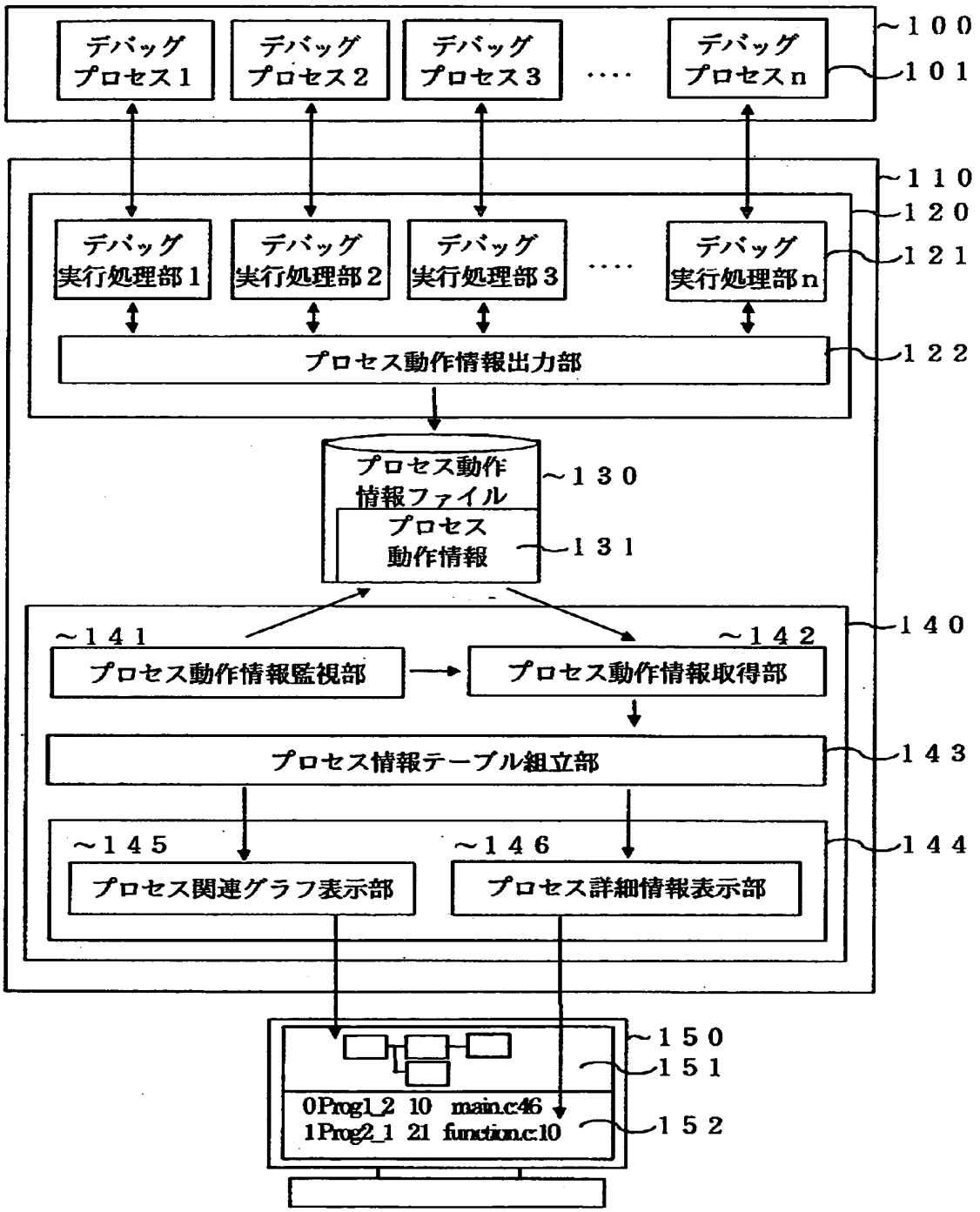
９２０　プロセス詳細情報からの強調表示操作

９４０　プロセス関連グラフとプロセス詳細情報の強調表示

【書類名】　図面

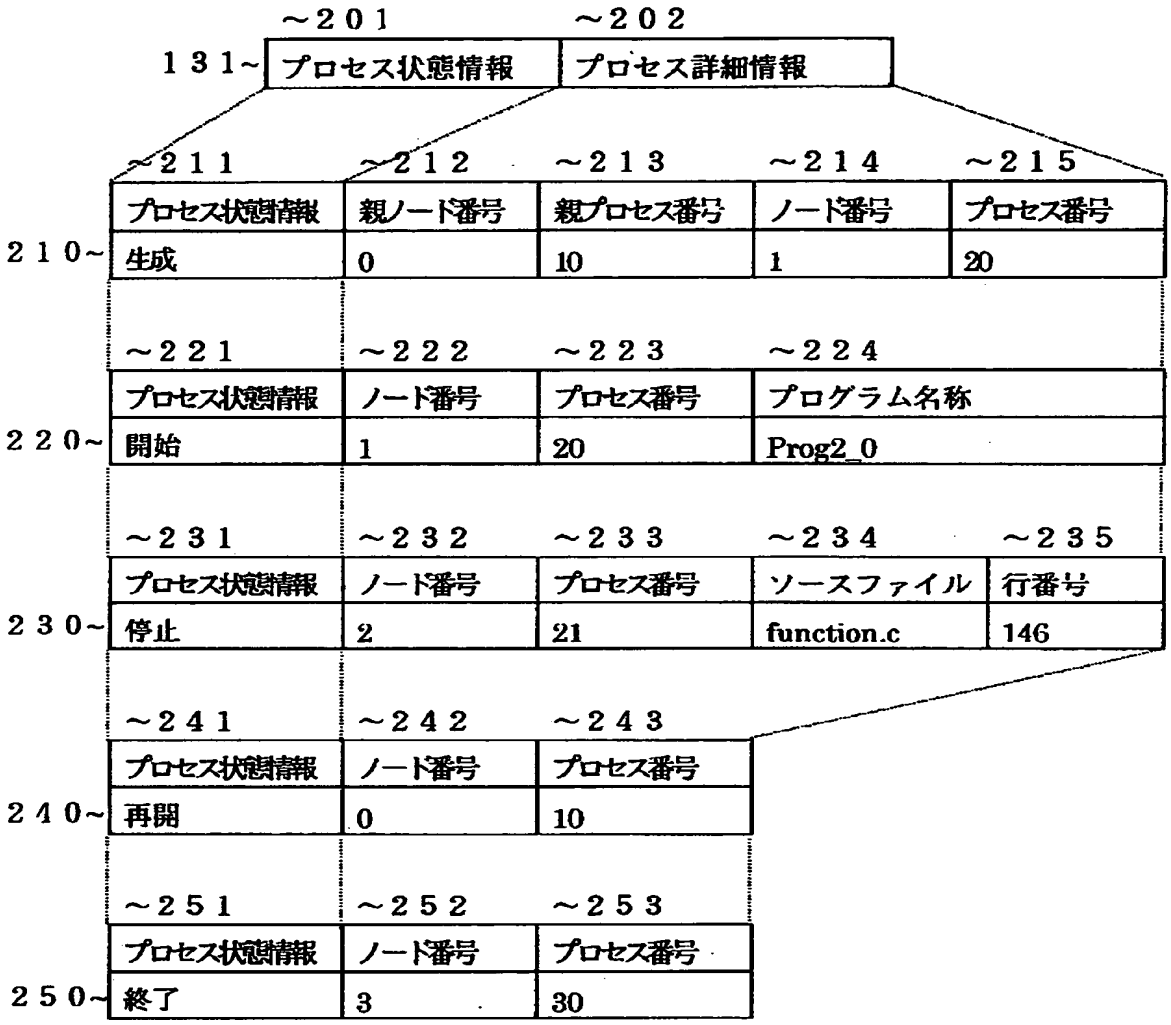
【図１】

図１


デバッグプロセス１
デバッグプロセス２
デバッグプロセス３
・・・・
デバッグプロセスｎ
１００
１０１
１１０
１２０
デバッグ実行処理部１
デバッグ実行処理部２
デバッグ実行処理部３
・・・・
デバッグ実行処理部ｎ
１２１
プロセス動作情報出力部
１２２
プロセス動作情報ファイル
１３０
プロセス動作情報
１３１
１４０
～１４１
プロセス動作情報監視部
～１４２
プロセス動作情報取得部
プロセス情報テーブル組立部
１４３
～１４５
プロセス関連グラフ表示部
～１４６
プロセス詳細情報表示部
１４４
１５０
１５１
0Prog1_2 10 main.c46
1Prog2_1 21 function.c10
１５２

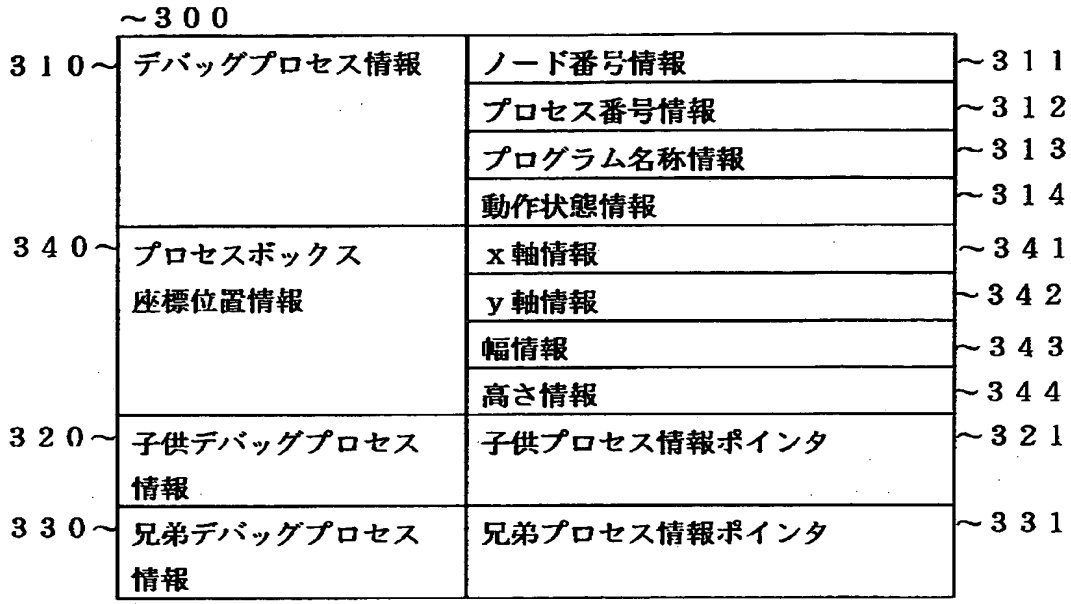
【図２】

図２


～２０１
～２０２
１３１～
プロセス状態情報
プロセス詳細情報
～２１１
～２１２
～２１３
～２１４
～２１５
プロセス状態情報
親ノード番号
親プロセス番号
ノード番号
プロセス番号
２１０～
生成
0
10
1
20
～２２１
～２２２
～２２３
～２２４
プロセス状態情報
ノード番号
プロセス番号
プログラム名称
２２０～
開始
1
20
Prog2_0
～２３１
～２３２
～２３３
～２３４
～２３５
プロセス状態情報
ノード番号
プロセス番号
ソースファイル
行番号
２３０～
停止
2
21
function.c
146
～２４１
～２４２
～２４３
プロセス状態情報
ノード番号
プロセス番号
２４０～
再開
0
10
～２５１
～２５２
～２５３
プロセス状態情報
ノード番号
プロセス番号
２５０～
終了
3
30

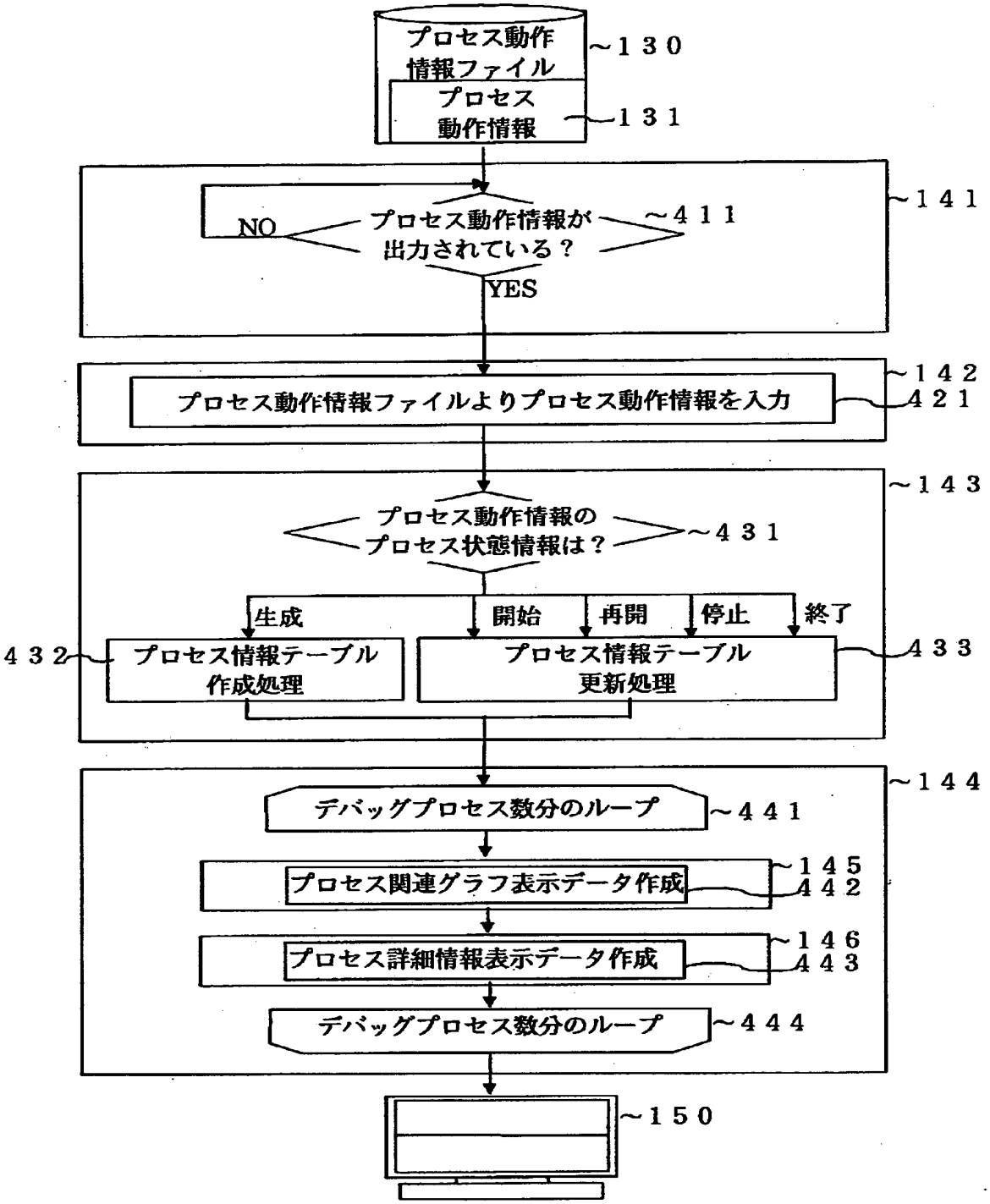
【図３】

図３


～３００
３１０～
デバッグプロセス情報
ノード番号情報
～３１１
プロセス番号情報
～３１２
プログラム名称情報
～３１３
動作状態情報
～３１４
３４０～
プロセスボックス座標位置情報
ｘ軸情報
～３４１
ｙ軸情報
～３４２
幅情報
～３４３
高さ情報
～３４４
３２０～
子供デバッグプロセス情報
子供プロセス情報ポインタ
～３２１
３３０～
兄弟デバッグプロセス情報
兄弟プロセス情報ポインタ
～３３１

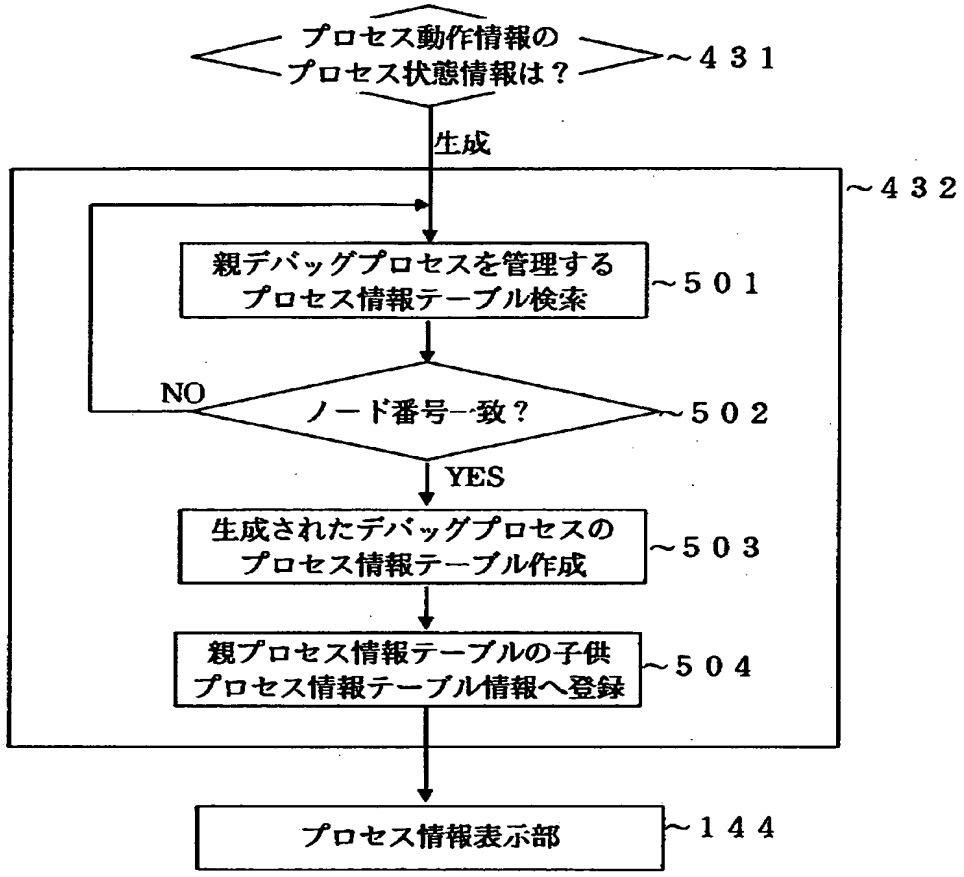
【図４】

図４


プロセス動作情報ファイル
～１３０
プロセス動作情報
～１３１
～１４１
NO
プロセス動作情報が出力されている？
～４１１
YES
～１４２
プロセス動作情報ファイルよりプロセス動作情報を入力
４２１
～１４３
プロセス動作情報のプロセス状態情報は？
～４３１
生成
開始
再開
停止
終了
４３２
プロセス情報テーブル作成処理
プロセス情報テーブル更新処理
４３３
～１４４
デバッグプロセス数分のループ
～４４１
～１４５
プロセス関連グラフ表示データ作成
４４２
～１４６
プロセス詳細情報表示データ作成
４４３
デバッグプロセス数分のループ
～４４４
～１５０

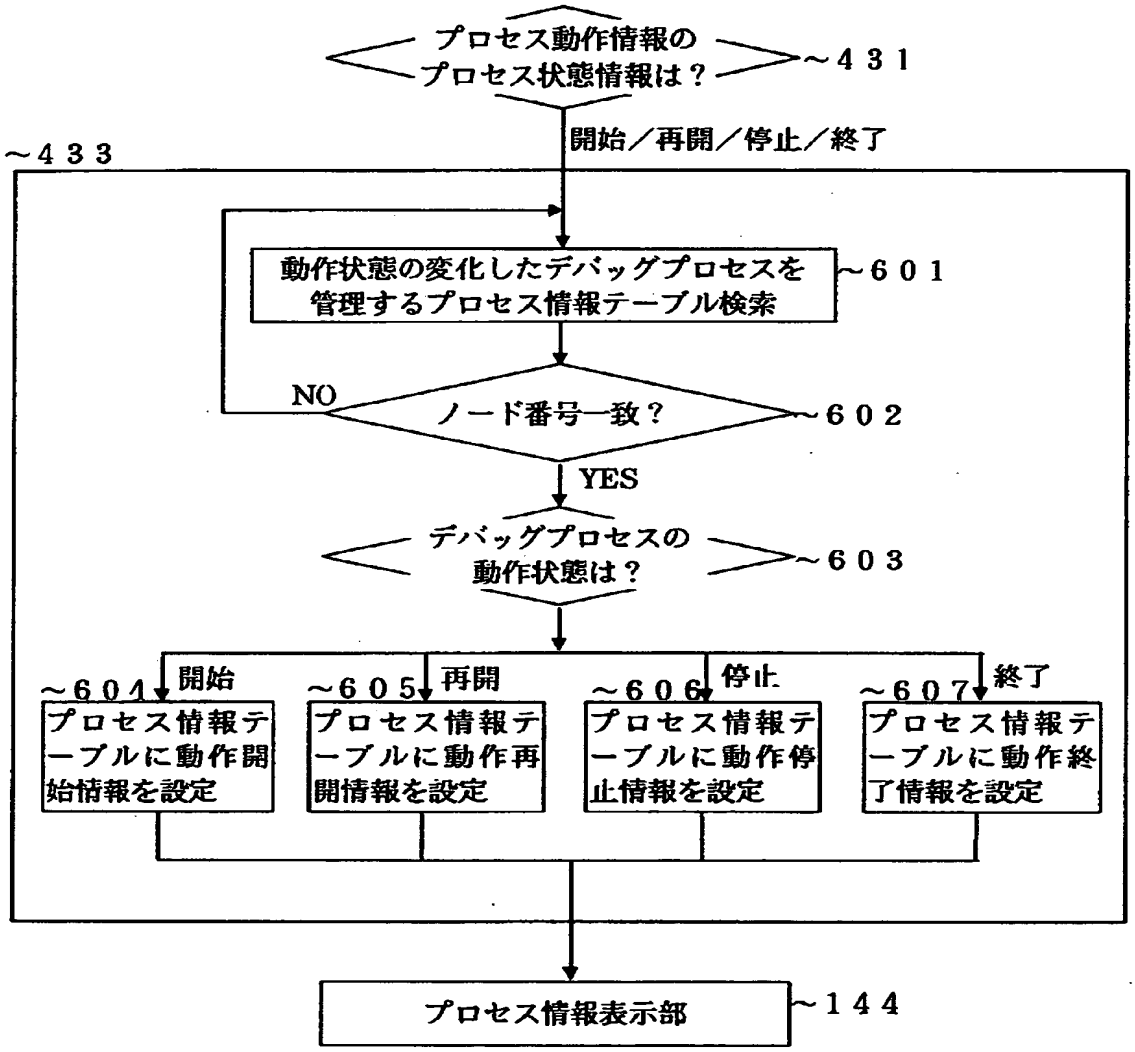
【図５】

図５


プロセス動作情報の
プロセス状態情報は？
～４３１
生成
～４３２
親デバッグプロセスを管理する
プロセス情報テーブル検索
～５０１
NO
ノード番号一致？
～５０２
YES
生成されたデバッグプロセスの
プロセス情報テーブル作成
～５０３
親プロセス情報テーブルの子供
プロセス情報テーブル情報へ登録
～５０４
プロセス情報表示部
～１４４

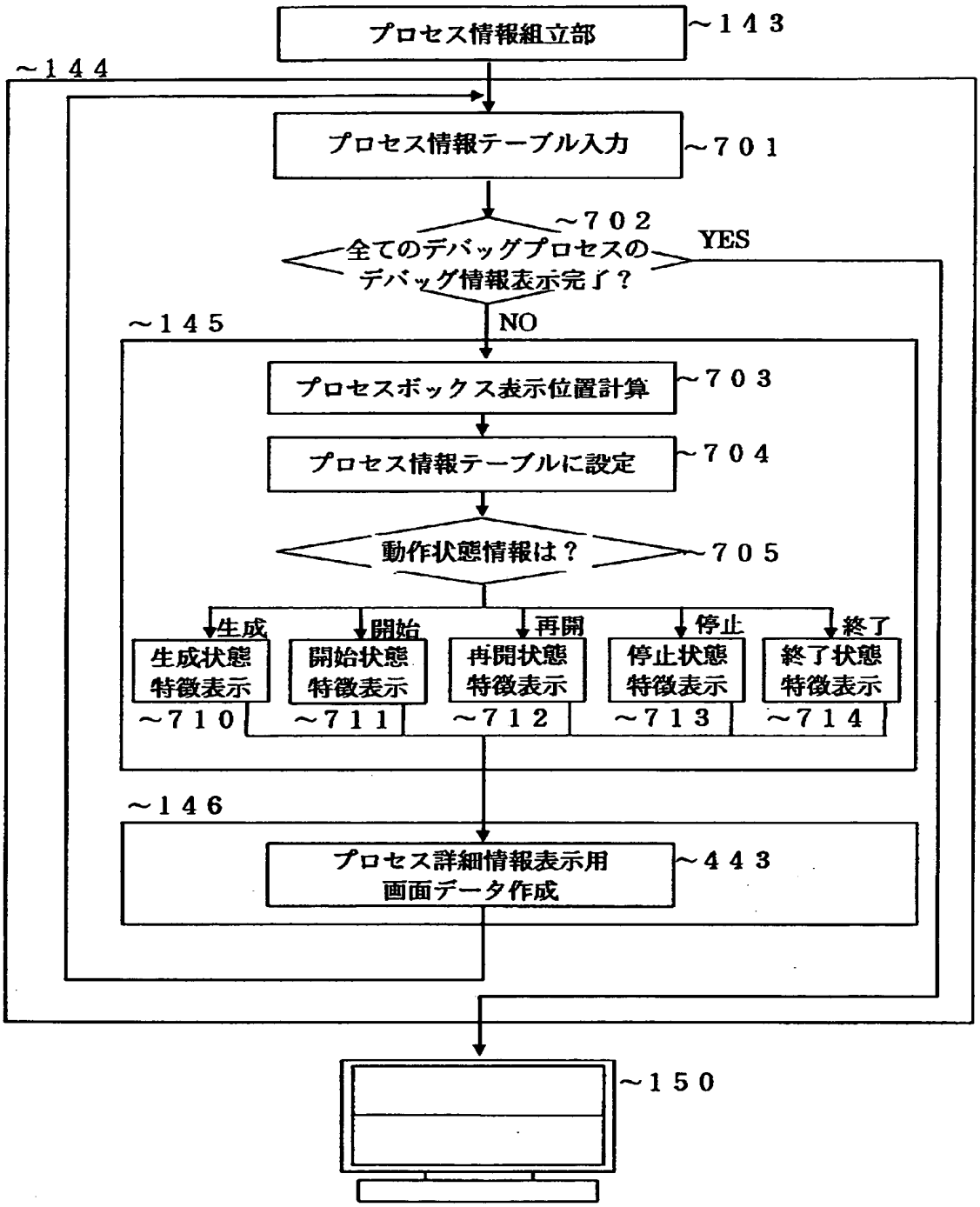
【図６】

図６


プロセス動作情報の
プロセス状態情報は？
～４３１
開始／再開／停止／終了
～４３３
動作状態の変化したデバッグプロセスを
管理するプロセス情報テーブル検索
～６０１
NO
ノード番号一致？
～６０２
YES
デバッグプロセスの
動作状態は？
～６０３
開始
～６０４
プロセス情報テーブルに動作開始情報を設定
再開
～６０５
プロセス情報テーブルに動作再開情報を設定
停止
～６０６
プロセス情報テーブルに動作停止情報を設定
終了
～６０７
プロセス情報テーブルに動作終了情報を設定
プロセス情報表示部
～１４４

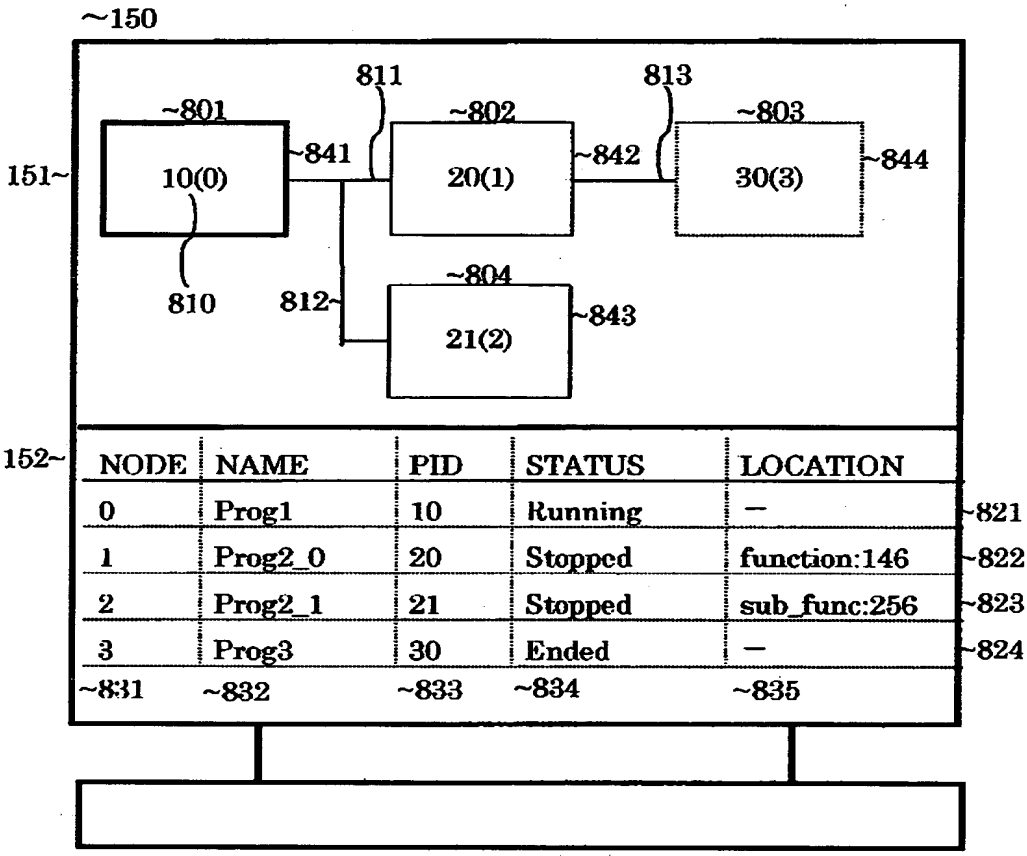
【図７】

図７


プロセス情報組立部
～１４３
～１４４
プロセス情報テーブル入力
～７０１
～７０２
全てのデバッグプロセスのデバッグ情報表示完了？
YES
NO
～１４５
プロセスボックス表示位置計算
～７０３
プロセス情報テーブルに設定
～７０４
動作状態情報は？
～７０５
生成
開始
再開
停止
終了
生成状態特徴表示
開始状態特徴表示
再開状態特徴表示
停止状態特徴表示
終了状態特徴表示
～７１０
～７１１
～７１２
～７１３
～７１４
～１４６
プロセス詳細情報表示用画面データ作成
～４４３
～１５０

【図８】

図８

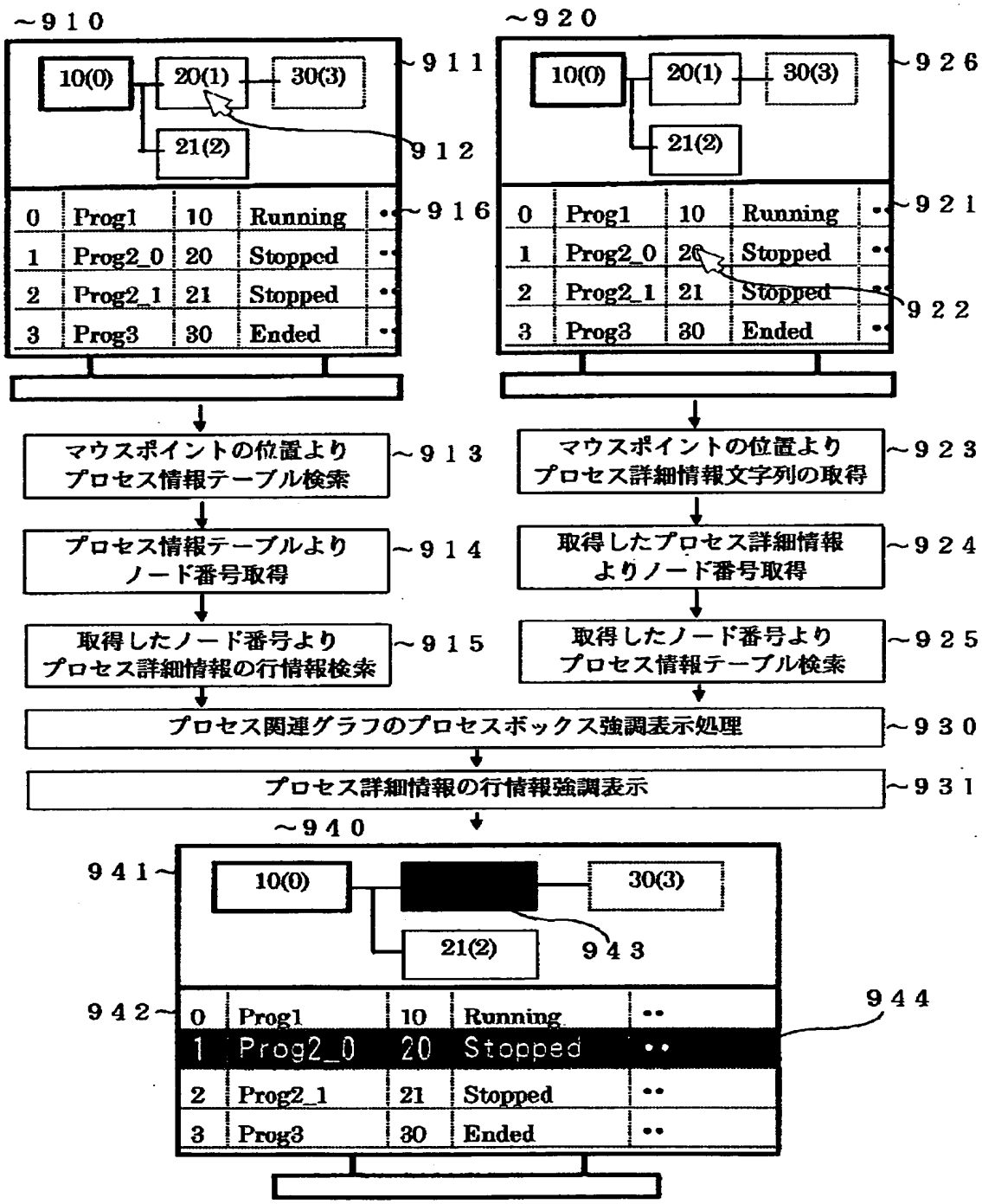


152~

NODE	NAME	PID	STATUS	LOCATION	
0	Prog1	10	Running	—	~821
1	Prog2_0	20	Stopped	function:146	~822
2	Prog2_1	21	Stopped	sub_func:256	~823
3	Prog3	30	Ended	—	~824
~831	~832	~833	~834	~835	

【図９】

図９



【書類名】　要約書

【要約】

【課題】

マルチプロセスプログラムを構成する複数のプロセスの動作状態に関連付けたデバッグ情報を、プログラムの動作と連動しながら特徴表示するデバッグ支援装置を提供する。

【解決手段】

デバッグプログラム１００を構成する複数のデバッグプロセス１０１の動作状態の変化を、デバッガ実行処理部１２１で監視し、デバッグプロセス１０１の動作状態に変化が生じた場合、プロセス動作情報出力部１２２はプロセス動作情報１３１を出力する。デバッガ情報表示部１４０は、プロセス動作情報を入力１４２し、デバッグプロセス１０１の親子、及び兄弟関係を視覚的に表現し、プロセスボックスを特徴表示する、プロセス関連グラフ１５１と、デバッグプロセス１０１の動作状態を詳細に表示するプロセス詳細情報１５２を同一画面に表示する。

【選択図】　図１

出 願 人 履 歴 情 報

識別番号　〔０００００５１０８〕

１．変更年月日　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所